RECEIVED
CENTRAL FAX CENTER
NOV 3 0 2006

Doc Code: AP.PRE.REQ

PTO/SB/33 (07-05)
Approved for use through xx/xx/200x. OMB 0651-00xx
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PRE-APPEAL BRIEF REQUEST FOR REVIEW

	Docket Number (Optional) NAI1P481/01.207.01	
I hereby certify that this correspondence is being transmitted via fascimile to the Commissioner for Patents, Alexandria, VA 22313-1450 to fax number (571) 273-8300. on November 30, 2006 Signature [illegible] Typed or printed name April Skovmand	Application Number 10/003,322	Filed 12/06/2001
	First Named Inventor Neil Andrew Cowle et al.	
	Art Unit 2131	Examiner Syed Zia

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a notice of appeal.

The review is requested for the reason(s) stated on the attached sheet(s).
Note: No more than five (5) pages may be provided.

I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest.
See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)

☑ attorney or agent of record. Registration number 41,429

☐ attorney or agent acting under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 ______

[illegible]
Signature

Kevin J. Zilka
Typed or printed name

408-971-2573
Telephone number

November 30, 2006
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☑ *Total of 1 forms are submitted.

REMARKS

The Examiner has rejected Claims 1-4, 9-16, 21-28, and 33-39 under 35 U.S.C. 103(a) as being unpatentable over Abu-Husein (U.S. Patent No. 6,895,506), in view of Nachenberg (U.S. Patent No. 5,826,013). Applicant respectfully disagrees with such rejection.

With respect to the independent claims, the Examiner has relied on Col. 3, lines 3-24 from the Nachenberg reference to make a prior art showing of applicant's claimed technique "wherein said computer program that is decrypted, loaded, and executed includes a malware scanning computer program."

Applicant respectfully asserts that the excerpt from Nachenberg relied upon by the Examiner merely teaches that the "[t]he PAM system (200) comprises a CPU emulator (210) for emulating the target program, a virus signature scanning module (250) for scanning decrypted virus code, and an emulation control module (220), including a static exclusion module (230) and a dynamic exclusion module (240), for determining how long each target file is emulated before it is scanned" (Col. 3, lines 6-12 – emphasis added). However, "emulating the target program" and "scanning decrypted virus code" does not disclose a technique "wherein said computer program that is decrypted, loaded, and executed includes a malware scanning computer program" (emphasis added) as claimed by applicant. Clearly, scanning decrypted virus code simply fails to even suggest a "malware scanning computer program" that is "decrypted, loaded, and executed" (emphasis added), in the manner as claimed by applicant.

Further, with respect to the independent claims, the Examiner has relied on Col. 3, lines 25-53 from the Nachenberg reference to make a prior art showing of applicant's claimed technique "wherein said malware scanning computer program is operable such that once executed, said malware scanning computer program scans said loader program for malware."

Applicant respectfully asserts that the excerpt from Nachenberg relied upon by the Examiner merely discloses that "the static exclusion module (230) examines the gross characteristics of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses" (Col. 3, lines 25-28). In addition, Nachenberg discloses that "at least some code from the decrypted static virus body (160) may be scanned" (Col. 3, lines 49-50). Clearly, teaching the examination of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses and that some code of decrypted static virus body may be scanned, as in Nachenberg, clearly fails to even suggest a technique "wherein said malware scanning computer program is operable such that once executed, said malware scanning computer program scans said loader program for malware" (emphasis added), as claimed by applicant.

In the Office Action dated 09/06/2006, the Examiner argued that "[a]n emulation control module includes a static exclusion module, a dynamic exclusion module, instruction/interrupt usage profiles for the mutation engines of the known polymorphic viruses, size and target file types for the viruses, and a table having an entry for each known polymorphic virus" and that "[a]s a result, cited prior art does implement and teach a system and method that relates to scanning computer code for viruses and for providing results pertaining to the viruses found by scanning of the initiation of execution of a computer program using a mechanism that seeks to protect the computer program from malicious alteration" (see Page 3).

Applicant respectfully disagrees with the Examiner's arguments and asserts that Nachenberg merely discloses that "[t]he PAM system (200) comprises a CPU emulator (210) for emulating the target program, a virus signature scanning module (250) for scanning decrypted virus code, and an emulation control module (220), including a static exclusion module (230) and a dynamic exclusion module (240), for determining how long each target file is emulated before it is scanned" (Col. 3, lines 7-13 – emphasis added). Further, Nachenberg discloses that "[o]nce the emulation phase has been terminated, scanning can begin on decrypted static virus body 160 or at least those parts decrypted by the first 1.5 million instructions" (Col. 9, lines 50-52 – emphasis added).

However, merely disclosing scanning the decrypted static virus body once an emulation phase is terminated, as in Nachenberg, simply fails to even suggest a technique "wherein said malware scanning computer program is operable such that once executed, said malware scanning computer program scans said loader program for malware" (emphasis added), as claimed by applicant. Clearly, a system in which instructions of a target program are emulated before the target program is scanned, as in Nachenberg, fails to meet and even *teaches away* from a technique where "said malware scanning computer program is operable such that once executed, said malware scanning computer program scans said loader program for malware" (emphasis added), in the manner as claimed by applicant.

In addition, with respect to the independent claims, the Examiner has relied on Col. 9, lines 50-67 from the Nachenberg reference to make a prior art showing of applicant's claimed technique "wherein, if said loader program is detected as being infected with said malware, then said malware scanning computer program is operable to repair said loader program or replace said loader program with a clean copy of said loader program."

Applicant respectfully asserts that the excerpt from Nachenberg relied upon by the Examiner merely discloses that "every time an emulated instruction writes to memory, the altered pages are tagged as containing modified data." Further, Nachenberg teaches that "[t]agged pages in virtual memory are scanned for signatures of static virus body." However, merely scanning tagged pages in virtual memory fails to even suggest a technique "wherein, if said loader program is detected as being infected with said malware, then said malware scanning computer program is operable to repair said loader program or replace said loader program with a clean copy of said loader program" (emphasis added), as claimed by applicant.

In the Office Action dated 09/06/2006, the Examiner argued that "[t]he computer virus detection module includes a CPU emulator for emulating a target program, and a virus signature scanning module for scanning decrypted virus code" (see Page 3). However, applicant respectfully asserts that merely "scanning [the] static virus body" and scanning "[t]agged pages in virtual memory…for signatures of [a] static virus body" (emphasis added) does not even suggest a program "operable to repair said loader program or replace said loader program with a clean copy of said loader program" (emphasis added), in the manner as claimed by applicant. Further, applicant respectfully asserts that Nachenberg merely discloses that "[i]n order to return control of the computer to program code 120 following execution of virus 130, CS, IP, SS, and SP of uninfected image 100 are retained by virus 130" (Col. 5, lines 36-39 – emphasis added). Clearly, the virus retaining uninfected fields in order to return control after the execution of the virus simply fails to even suggest a program "operable to repair said loader program or replace said loader program with a clean copy of said loader program" (emphasis added), in the manner as claimed by applicant.

Applicant further notes that the prior art is also deficient with respect to the dependent claims. For example, with respect to Claim 10 et al., the Examiner has relied on Col. 8, line 65 - Col. 9, line 15 from the Abu-Husein reference to make a prior art showing of applicant's claimed technique "wherein said computer program is operable to terminate said loader program when said computer program is triggered to execute by said loader program."

Applicant respectfully asserts that the excerpt from Abu-Husein relied upon by the Examiner merely discloses that '"context switches" ... may occur ... when a Process 16 must call another Process 16 for an operation or has been completed or when the processor "time slice" allocated to a Process 16 has ended and the processor is to be allocated to the execution of a different Process 16' (Col. 9, lines 10-15 – emphasis added). Clearly, the concept of "time slices" and "context switches" simply fails to even suggest a technique "wherein said computer program is operable to terminate said loader program when said computer program is triggered to execute by said loader program" (emphasis added), as claimed by applicant.

In the Office Action dated 09/06/2006, the Examiner failed to specifically respond to applicant's arguments and simply dismissed them as non-persuasive. Applicant emphasizes that Abu-Husein discloses that "[d]uring a context switch, the execution of the currently executing Process 16 is suspended and the storage space in Memory 12C is reassigned by the Memory Manager 30A to accommodate the programs and data of the new Process 16" (Col. 9, lines 16-19 – emphasis added). Clearly, Abu-Husein's disclosure that the Process 16 is suspend during a context switch, simply fails to even suggest that the "computer program is operable to terminate said loader program when said computer program is triggered to execute by said loader program" (emphasis added), in the manner as claimed by applicant.

Further, with respect to Claim 11 et al., the Examiner has relied on Col. 9, lines 16-50 from the Abu-Husein reference to make a prior art showing of applicant's claimed technique "wherein said loader program is operable to load said computer program into a memory space within said computer memory separate from a memory space used by said loader program." Applicant respectfully asserts that excerpt from Abu-Husein relied upon by the Examiner merely discloses a method of performing a "context switch." Specifically, Abu-Husein discloses a context switch where "the execution of the currently executing Process 16 is suspended and the storage space in Memory 12C is reassigned by the Memory Manager 30A to accommodate the programs and data of the new Process 16."

Additionally, Abu-Husein teaches that the context of the current process is saved 'to allow the execution of the Process 16 to be resumed at the point it was suspended when the

Process 16 is "switched" or "swapped" in and again becomes the Process 16 currently being executed.' However, merely disclosing a context switch fails to even suggest a technique "wherein said loader program is operable to load said computer program into a memory space within said computer memory separate from a memory space used by said loader program" (emphasis added), as claimed by applicant.

In the Office Action dated 09/06/2006, the Examiner failed to specifically respond to applicant's arguments and simply dismissed them as non-persuasive. Applicant emphasizes that Abu-Husein merely discloses that "Program Loader/Decryption Mechanism (Load/Decrypt) 28, which may be implemented, for example, as a program controlled Process 16 executed by Processor 12A" (Col. 6, lines 53-57 – emphasis added). Further, Abu-Husein discloses that "Load/Decrypt 28 is similar to a conventional loading utility in responding to calls, commands or requests for the execution of a program or a program module or function by operating in conjunction with a Memory Manager 30A to read the corresponding program code from Storage 12E and to write the program code into Memory 12C at locations selected and managed by Memory Manager 30A for execution by Processor 12A" (Col. 6, lines 60-67 – emphasis added). However, the disclosure that the Load/Decrypt, which is a program controlled process executed by the Processor, operates in conjunction with Memory Manager to read the code from Storage and write the code into Memory for execution by the Processor, as in Abu-Husein, clearly fails to even suggest a technique "wherein said loader program is operable to load said computer program into a memory space within said computer memory separate from a memory space used by said loader program" (emphasis added), as claimed by applicant.

A notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.